Region-Based Routing: A Mechanism to Support Efficient Routing Algorithms in NoCs

Andres Mejia, Maurizio Palesi, José Flich, Shashi Kumar, *Senior Member, IEEE*, Pedro López, Rickard Holsmark, and José Duato, *Member, IEEE*

***Abstract*—An efficient routing algorithm is important for large on-chip networks [network-on-chip (NoC)] to provide the required communication performance to applications. Implementing NoC using table-based switches provide many advantages, including possibility of changing routing algorithms and fault tolerance, due to the option of table reconfigurations. However, table-based switches have been considered unsuitable for NoCs due to their perceived high area and power consumption. In this paper, we describe the region-based routing (RBR) mechanism which groups destinations into network regions allowing an efficient implementation with logic blocks. RBR can also be viewed as a mechanism to reduce the number of entries in routing tables. RBR is general and can be used in conjunction with any adaptive routing algorithm. In particular, we have evaluated the proposed scheme in conjunction with a general routing algorithm, namely segment-based routing (SR) and an Application Specific Routing Algorithm (APSRA) using regular and irregular mesh topologies. Our study shows that the number of entries in the table is significantly reduced, especially for large networks. Evaluation results show that RBR requires only four regions to support several routing algorithms in a 2-D mesh with no performance degradation. Considering link failures, our results indicate that RBR combined with SR is able to tolerate up to 7 link failures in an 8×8 mesh. RBR also reduces area and power dissipation of an equivalent table-based implementation by factors of 8 and 10, respectively. Moreover, the degradation in performance of the network is insignificant when using APSRA combined with RBR.**

***Index Terms*—Application-specific routing, deadlock-free routing, networks-on-chip (NoC), region-based router (RBR), routing algorithms, router architecture, table-based router.**

I. Introduction

SYSTEMS-ON-CHIP (SoCs) used in embedded systems are becoming increasingly large, complex, and heterogeneous. A typical SoC is composed of several general-purpose cores (like processor and memory cores) together with application-specific cores and special circuitry. In an SoC, the design of an efficient interconnection structure is an important task since it not only determines its computing performance but also may affect its power consumption, design time, test time, and fault tolerance. Due to a very high nonrecurring engineering cost of manufacturing an SoC, researchers and industry have been looking for a reusable, scalable, and structured solution to the interconnection problem. This has led to the proposal of packet switched networks for on-chip communication or network-on-chip (NoC) [1], [2]. In fact, many ideas and terminology for NoCs have been borrowed from the area of interconnection networks for massively parallel processors and clusters.

A. Network Topologies and Routing Algorithms

The network topology and routing algorithm are the two most important features that influence the network performance, cost, and power consumption. Different topology and routing proposals have been made for NoCs [1]–[4]. Regarding the topology, fixed tile size-based mesh topology is favored by many researchers because of its layout efficiency on 2-D surface and the resulting predictable and good electrical properties of the signals. Mesh topology also makes addressing of the cores quite simple during routing. However, a physically homogeneous network will not always be possible in many cases. One case is when incorporating cores of different sizes in the network. A solution to this problem is to allow cores larger than the tile size to occupy a rectangular area covering multiple tiles [2]. However, the final layout disturbs the regularity of the topology. Another reason where the regularity of the topology is lost is when fabrication faults appear in the form of defective core nodes, wires, or switches. In these cases, functional parts of the chip could be still used but its topology may no longer be regular. The chips manufactured using deep submicrometer technology also have chances of process and layout variations resulting in large crosstalk. This can result in glitch or delay faults which might make communication using certain network links error-prone with data error rates higher than the acceptable limits [6]. In these cases, it may be advisable to disable such links, thus, resulting in a non-regular topology.

On the other hand, the routing algorithm determines the path that each packet follows between a source-destination pair. Routing is deterministic if only one path is provided, or adaptive if several paths are available and dynamically selected at switches. Generally, an adaptive routing algorithm is preferable since it has the potential of achieving higher performance (low latency, high throughput, and fault tolerance). Routing strategies can also be classified as source or distributed. In source routing, the source node stores the entire path in the packet header. Since the header itself must be transmitted through the

Manuscript received December 08, 2007; revised April 02, 2008. Current version published February 19, 2009. This work was supported by the HIPEAC 1, Cluster Research on High Performance Interconnection Networks for Embedded Applications and by CONSOLIDER-INGENIO 2010 under Grant CSD2006-00046.

A. Mejia, J. Flich, P. López, and J. Duato are with Universidad Politécnica de Valencia, Valencia 46022, Spain (e-mail: andres@gap.upv.es; jflich@gap.upv.es; plopez@gap.upv.es; jduato@gap.upv.es).

M. Palesi is with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Facolt di Ingegneria, University di Catania, Catania 95125, Italy (e-mail: mpalesi@diit.unict.it).

S. Kumar and R. Holsmark are with Jönköping University, Jönköping SE-551 11, Sweden (e-mail: shashi.kumar@jth.hj.se; rickard.holsmark@jth.hj.se).

Digital Object Identifier 10.1109/TVLSI.2008.2012010

network, it reduces the effective network bandwidth (specially with short packets). In distributed routing, the packet header contains the address of the destination. Each switch computes the next link that will be used while the packet travels across the network. Distributed routing is preferred for NoCs as the packet header is reduced and more flexibility and adaptivity can be achieved.

Distributed routing can be implemented using two distinct methods. The first method is called algorithmic routing and is suitable for on-chip and off-chip networks with regular topologies. In this method, a finite-state machine (FSM) is used for computing the routing option as a function of the current and destination node addresses along with possibly other information like status of output ports, network traffic, packet priority, etc. The implementation is very efficient in both area and speed, but it is topology and routing algorithm dependent. For instance, the Dimension Order Routing (DOR) Algorithm [7] for meshes routes packets by comparing at each switch the row and column (in the mesh structure) of the destination switch. First, dimension X is traversed and then dimension Y.

The second method uses a table which stores the output port(s) that should be used for each destination end node [8]. The main advantage of this approach is that the same generic design can be reused for implementing any network topology and routing algorithm. It is also much easier to incorporate fault tolerance with this approach as compared to the algorithmic approach.

B. Motivations and Contribution

Although there are some proposals to use table-based switches for building on-chip networks [8], [9], [10] such designs are not suitable because of the perceived higher cost, higher power consumption, and lack of scalability when implementing the tables in SRAM memories. Indeed, as the system increases in size, the memory requirements for building such routing tables also increase, thus requiring extra memory resources, exhibiting increased access latencies and power consumption levels.

Therefore, one open problem is how to efficiently implement the routing algorithm in a NoC with possibly a non-regular topology (due to heterogeneous cores, fabrication faults, or crosstalk). On one hand, the FSM approach can not be used due to the irregularity of the network. On the other hand, the table-based approach does not scale with system size. In this paper, we propose a mechanism referred to as the region-based routing mechanism (RBR) that will solve this problem. In particular, RBR will exhibit very low and constant memory and area requirements regardless of NoC size. Also, the mechanism will allow the use of any deterministic or partly adaptive routing algorithm in any 2-D mesh network with any derived irregularity (link or switch failure).

Another issue is the selection of a proper routing algorithm for a particular SoC. Many routing algorithms exist in the literature for off-chip networks that can be applied to NoCs. However, there are two important differences in current NoCs that should be considered when selecting the routing algorithm. The first one is that although the network will be irregular, usually the failure pattern or topology, will be known in advance. A common case could be a regular network (2-D mesh) with some links/switches disabled or failed, or some cores with larger sizes. Although many topology-agnostic routing algorithms exist, only one considers the underlying topology when computing the paths. This is the case of the segment-based routing (SR) [11] algorithm. By considering the underlying topology, SR benefits from the remaining regularity of the network and achieves relatively higher performance. SR is based on a segmentation process of the network and a placing of routing restrictions at each segment in order to guarantee deadlock freedom and connectivity. However, many patterns for searching segments exist and they may impact on the final performance depending on the topology. In this paper we apply the SR algorithm to NoCs in the context of the RBR mechanism.

Most of the routing algorithms proposed in the literature are general purpose and have been designed to handle worst case communication patterns, they assume the possibility of any node to communicate with any other node at any time. However, in an NoC system specialized for a specific application or a set of concurrent applications, the communication graph is known in advance. After the applications have been mapped and scheduled on the NoC system, we even have information about the set of communication transactions which are concurrent and others which are not concurrent. This information can be used to design highly adaptive deadlock-free routing algorithms. Such algorithms are called Application Specific Routing Algorithms (APSRAs) [12]. The most natural way to implement such routing algorithms is to use forwarding table-based switches. In this paper we also evaluate the APSRA methodology in the context of the RBR mechanism.

To sum up, in this paper, we describe the RBR mechanism for reducing the cost of a table-based switch. RBR is capable of handling irregular networks. We, then, evaluate the performance and fault tolerance of RBR in conjunction with two important routing algorithms, namely SR algorithm [11] and APSRA [12].

The rest of this paper is organized as follows. In Section II, we present related work in routing for NoCs. Later, in Section III, we introduce the RBR mechanism, by describing the required hardware architecture and analyzing the algorithm used for computing regions. In Sections IV and V, SR and APSRA methodologies are described and evaluated when applied to RBR. In Section VI, both methodologies (SR and APSRA) are evaluated and compared in view of resources needed for the RBR mechanism. We also provide area and energy estimations and compare them against a table-based switch implementation. Finally, in Section VII, we conclude our work and propose directions for future research.

II. Related Work

Different routing algorithms have been proposed for regular topologies. The most well known routing algorithm is dimension order routing (DOR) also known as XY [7]. This routing algorithm forwards every packet through each dimension at a time, following an established order. Notice that DOR routing does not tolerate a single link failure since some destinations would become unreachable. Also, different partly adaptive routing algorithms have been proposed for regular topologies.

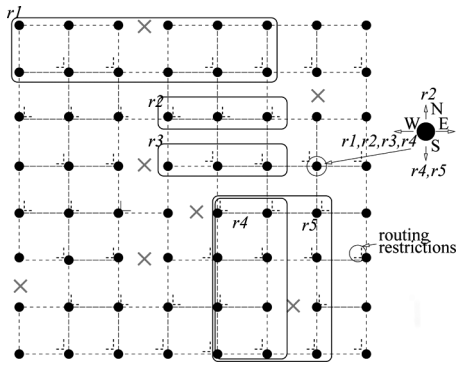


Fig. 1. Example of region definitions.

The turn model [13] avoids deadlock by restricting some turns within the network. Based on this approach different (west-first, north-last, negative-first) layouts have been proposed for two-dimensional meshes. Unfortunately, the degree of adaptiveness provided by these routing algorithms is highly unbalanced across the network, which in some cases results in poor performance. The Odd-Even (OE) Model [14] was proposed to solve this issue. This routing algorithm combines different turn prohibitions at odd and even columns, thus providing a more balanced distribution of adaptiveness throughout the network. Unfortunately, none of these routing algorithms provide valid solutions for failures in general.

To deal with irregular topologies we can use a topology-agnostic routing algorithm. Different topology-agnostic routing algorithms can be found in the literature, all of them devoted to off-chip interconnection networks. Examples are smart-routing (SMART) [20], Depth First Spanning Tree (DFS) [21], Up*/Down* (UD) [22], Segment-based Routing (SR) [11], Flexible Routing (FX) [23], and Left-Up-First Turn (LTURN) [24]. All of them are deterministic in the sense that only one path is provided for each source-destination pair. However, some of them may offer alternative paths (partly adaptive routing algorithms). Contrary to all the proposals, SR benefits from the underlying topology by using a segmentation process. In particular, SR exploits the underlying regularity implicit in the topology (for instance a 2-D mesh). Some topology agnostic deadlock free routing algorithms which use virtual channels have also been proposed [16] and [17].

Alternatively, fully adaptive routing algorithms can also be applied to NoCs. They allow the formation of cycles in the channel dependency graph (CDG). However, acyclic escape paths must be provided through additional resources (typically virtual channels). Although they usually obtain much higher performance, they have two important drawbacks. The first one is out-of-order delivery of packets, which for some applications may be unacceptable. The second drawback is that fully adaptive routing algorithms are designed to work in well known regular networks, thus being implemented in FSM. Thus, in non-regular networks, extra logic, area, and complexity would be required.

Regarding routing implementation in NoCs, to the best of our knowledge there are two significant works focused in reducing the routing tables and/or routing through irregular networks. In [10], Bolotin *et al.* presented a technique for minimizing the size of the routing table based on a combination of a predefined routing function and a minimal deviation routing table. This approach takes into account the communication traces of the application in order to achieve considerable reduction of the routing cost. Unfortunately, the cost of routing increases with the irregularity of the topology and it does not state clearly how deadlock freedom is guaranteed. Also, in [8], a method for switch table compression has been proposed. Although it helps in reducing the size of the routing table, it is aimed only for application-specific routing. Also, the mechanism is proposed for minimal path routing, thus, it is not designed to work in networks with some failed links/switches.

III. Region-Based Routing

In this Section, we provide a detailed description of the RBR mechanism. First, we provide a general overview and the foundations of the mechanism. Later, we describe the circuitry required for implementing RBR in NoC switches. Then, we analyze the algorithm used for computing the regions and how the hardware setup is performed. It has to be mentioned that computation of regions is performed offline, thus, not influencing network performance. Once computed, region information is downloaded to switches and the network is set for normal operation.

A. Basic Idea and Overview

The basic idea in the NoC context is to group destinations into regions for taking a routing decision at any switch. This idea is especially natural for networks which have their physical layout on a two dimensional surface matching with their logical topology. The most popular topology for NoC architectures, namely 2-D mesh topology has this property. It can be shown that for such topologies, the number of required regions is either constant or grows very slowly with network size. All the destinations within a region are equivalent for routing purposes. Therefore, a few entries in the routing table are sufficient for storing admissible output port options for all the nodes in the region.

Roughly, at each switch each region will be defined by all the destinations that can be reached using the same set of output ports. As an example, Fig. 1 shows an 8×8 mesh network with seven link failures. Ports at every switch are labeled as N (north), E (east), W (west), S (south) and I (internal). The Figure also shows the set of bidirectional routing restrictions defined by the applied routing algorithm (SR has been applied for this case). A routing restriction is defined by two consecutive links in a given switch. Both links can not be used consecutively by any packet in order to guarantee deadlock freedom.

In Fig. 1, some switches have been grouped into regions $(r1,\ r2,\ r3)$. Regions are defined for the switch highlighted with a circle (this switch will be referred to as the *reference* switch). All the packets arriving to the *reference* switch (or being generated at its local port) addressed to any destination included in region $r1$ necessarily need to use the W output port at the *reference* switch. Notice that using the N port, requires crossing a forbidden routing restriction or use a non-minimal route which may be inefficient. The same happens when using

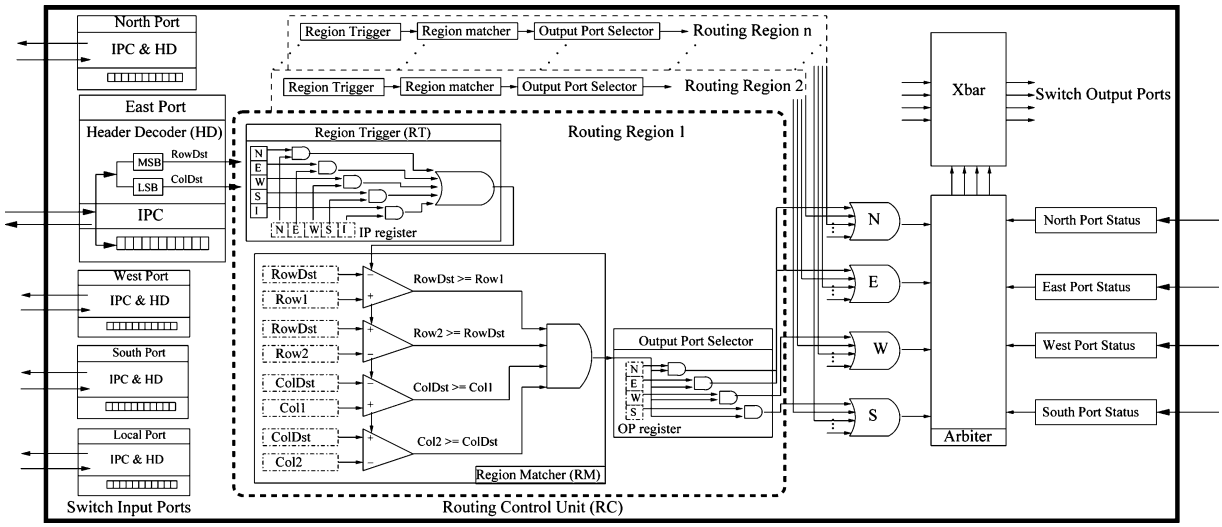


Fig. 2. Proposed hardware implementation of the RBR mechanism.

the E and S ports. Therefore, at the *reference* switch, region $r1$ includes destinations that can be reached only through the W port. Also, there are other destinations that can be reached using only the W port (i.e., switches within region $r3$), however, the mechanism relies on defining rectangular regions for the sake of implementation, thus, two regions ($r1$ and $r3$) are defined for the same output port.

Notice that regions are defined by taking into account the routing restrictions provided by the applied routing algorithm. By doing this, we ensure that the final routing with regions will not lead to deadlock as packets will follow the routing restrictions.

An interesting property of the mechanism is that regions can be defined for more than one output port, thus allowing the adaptiveness inherent to the applied routing algorithm. For instance, region $r2$ is defined by those destinations that can be reached either using the N or W ports at the *reference* switch.

The input port from which the packet arrived to the *reference* switch must be also taken into account when defining regions. For instance, at first sight, we can use ports W and S to reach region $r4$. However, W output port should only be used when the packet does not reach the *reference* switch through the N port, as it would cross a routing restriction at the *reference* switch and potentially would lead to deadlock. Therefore, regions must be defined based on the set of destinations, the output ports and the input ports. Notice also that the set of destinations of two regions can be overlapped, but they will differ in the set of input ports.

To summarize, a set of regions is computed at every switch. Each region is defined by the possible input ports used by the packets, a subset of the output ports that may be used and the potential destinations that can be reached. Additionally, as regions are defined by rectangular boxes of switches in a 2-D mesh network, we can notate a region for switch x as $R_x(\text{iport list}, \{sw_{\text{ref1}}, sw_{\text{ref2}}\}, \text{oport list})$, where sw_{ref1} is the top left most switch and sw_{ref2} is the bottom right most switch.

Whenever a packet arrives into a switch, all the regions must be inspected in order to detect which are the regions suitable for routing the packet. The routing unit, thus, selects the set of output ports provided by the matched regions and delivers all the possible output ports to the arbiter in charge of selecting the most convenient one.

B. Hardware Description

In this Section, we describe a simple implementation of RBR for a $N \times N$ mesh. Fig. 2 shows the proposed implementation. The mechanism requires nearly the same functional blocks as any other NoC switch, that is, from left to right in Fig. 2: the input ports, the routing control unit (routing function), the arbiter and scheduler (selection function), and the crossbar and the output ports.

The input port for wormhole switches generally requires two different blocks, an input port controller (IPC) that manages the input buffer and transmits the status information to neighboring nodes, and a header decoder (HD) that decodes the header information of every packet. The packet header should be as compact as possible in order to keep the minimum overhead. Among other information it contains the destination identifier of the packet. In our proposal, an absolute addressing is performed, and the packet ID identifies the X and Y coordinates of the destination [the MSBs indicate the row (RowDst) and the LSBs indicate the column (ColDst)] as indicated in the east port of the router in Fig. 2. Once decoded, the coordinates of the destination are compared with the coordinates of the current switch (not shown in Fig. 2). If equal, the packet is delivered to the internal port, otherwise, the coordinates are sent to the routing control unit shown in the center of Fig. 2.

The routing control (RC) unit is made of different logic modules, one for each possible region defined in the switch. Modules are composed by three different submodules, the region matcher, region trigger, and output port selector. Each module has six registers, the input port (IP) register with one bit per input port (NEWSI; North, East, West, South, and Internal),

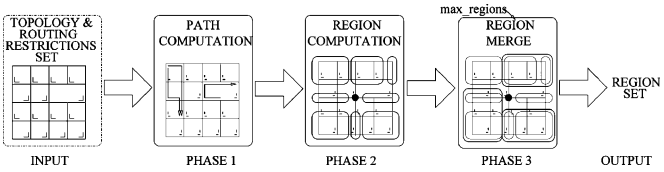


Fig. 3. Phases in the computation process of regions.

the ROW_1, COL_1, ROW_2, COL_2 registers with $\mathrm{Log}(N)$ bits each (these registers contain the coordinates of the top left most switch and the bottom right most switch of the region), and the output port (OP) register with one bit per output port (NEWS) (this register does not take into account the internal port as the packets going to the current switch have previously been delivered). These registers must be programmed before routing any packet at network boot time. The way register values are computed is described in Section III-C.

Once the packet header has been decoded it is passed to the RC unit together with the input port identifier where the packet arrived to the switch. At the RC unit this information is compared with the predefined regions. In order to reduce power consumption a preselection of regions is made according to the IP registers. That is, we discard checking the regions whose input port registers do not match with the input port of the packet being routed. To do this, the RT (region trigger) unit matches the input port of the packet with the input port register of every region. The RT unit consists of five AND gates (one per input port) and a single OR gate. The output of the RT unit is a signal that triggers the remaining logic for the region.

After checking the input port, and only on success, the logic determines if the destination is within the boundaries defined by the region. To achieve this, the row and column of the packet's destination are compared with the contents of ROW_1, COL_1, ROW_2, and COL_2 registers. For this, four magnitude comparators are used at the region matcher (RM) unit. If all the comparisons are true, the destination of the packet is within the region and therefore, the output ports defined for the region must be considered for routing purposes. Thus, the logic selects the output ports previously introduced in the OP register. Notice that the implementation allows different output ports to be selected from the same region (adaptivity).

Once all regions have been evaluated in parallel, all the selected output ports from all the regions that matched the packet are ORed and passed to the arbiter. The arbiter may choose one output port based on different criteria. Then, the packet will be routed to the next switch until arriving to its destination.

As shown, the control unit of RBR can be implemented using a very simple and fast logic block as it is shown within the dotted square in Fig. 2. Note that RT and RM units have been serialized (one is performed after the other). Although such a decision will increase latency, it will reduce energy consumption. Thus, there is a tradeoff between the routing time and power consumed.

C. Region Computation Algorithm

The algorithm for computing regions is divided into sequential phases. Fig. 3 shows the different phases. Initially, as input parameters, the algorithm receives the topology of the network and a set of routing restrictions. The network can be a 2-D mesh

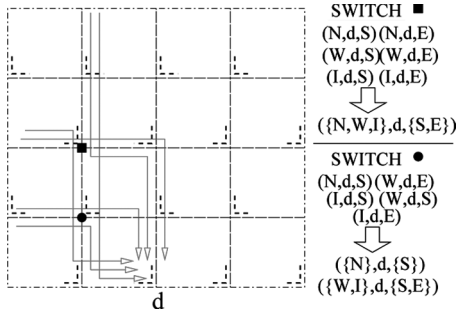


Fig. 4. Examples of routing paths and routing options. A routing option is indicated by the input port, destination, and output port.

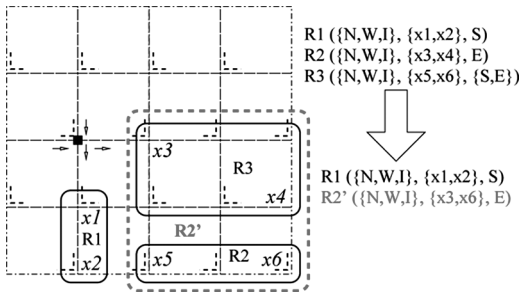


Fig. 5. Examples of some computed regions. A region is indicated by the input ports, set of destinations, and output ports.

of any size and with or without failures. As an example, Fig. 3 shows a 2-D mesh with a link failure.

The set of routing restrictions depends on the applied routing algorithm. Notice that when considering the set of routing restrictions all the paths can be computed in a later phase. Thus, routing is not restricted at this stage.

In a first phase, the algorithm computes the possible set of routing paths for every pair of nodes. This is a challenging problem since the number of possible paths can be extremely high. For this, the algorithm first tries to compute all the minimal paths for every source-destination. If, for a particular pair of nodes, there is no valid minimal path (in case the topology is irregular), then it searches for paths allowing an increasing number of misrouting hops. Once a non-minimal valid path is found, the algorithm switches to the next pair of nodes. Thus, the algorithm finds all the possible minimal paths and, when necessary, finds a unique non-minimal path. As an example, Figs. 4 and 5 show different paths and regions computed for the 2-D mesh. All the paths have in common the destination (switch d).

The routing paths are computed and stored in a distributed way (see Fig. 6 and Table I for a description of the used functions). In particular, the algorithm packs all the routing info into a set of routing options. Also, routing options are grouped per switch where they are applied. Each routing option $(\mathrm{ip},\ \mathrm{dst},\ \mathrm{op})$ is defined by an input port (ip), a destination (dst), and an output port (op), and indicates that a packet at the switch where the routing option is defined, that is coming through the input port ip, and going to destination dst may take output port op. For instance, in Fig. 4, routing options for switches plotted as a box and as a circle are shown. For instance, (N, d, S) at the box switch indicates that a packet

TABLE I
DESCRIPTION OF THE FUNCTIONS USED IN THE PSEUDOCODE SHOWN IN FIGS. 6–8

Function	Description
GetPaths(src, dst, T, RR)	Returns all the minimal paths for the communication from node *src* to node *dst*, for the topology T and considering routing restrictions *RR*. If no minimal paths are present, it returns the first non-minimal path.
InputPort(s, p)	Returns the input port identifier of switch s in which the path p pass through.
OutputPort(s, p)	Returns the output port identifier of switch s in which the path p pass through.
Packet($RO[s], port\_type$)	Packetizes routing options for switch s per $port\_type \in \{INPUT\_PORT, OUTPUT\_PORT\}$ port.
GetOutputCombinations(s)	Returns all the combinations of the output ports of switch s.
GroupDestinations($i, o, RO[s]$)	Groups destinations reachable through the output port o at the switch s and coming from the set of inputs i.
CanBeMerged(R_1, R_2)	Returns true if regions R_1 and R_2 can be merged, otherwise it returns false.
Merge(R_1, R_2)	Returns the region obtained by the merging of regions R_1 and R_2.

```
1  PathsComputation(in: Topology T,
2                              RoutingRestrictions RR;
3                   out: RoutingOptions RO[]){
4    for (src,dst ∈ T, src ≠ dst) {
5      paths = GetPaths(src,dst,T,RR);
6      for (path ∈ paths, switch ∈ path)
7        RO[switch] = RO[switch] ∪ {InputPort(switch,path),dst,
8                                  OutputPort(switch,path)};
9    }
10   for (switch ∈ T) {
11     Pack(RO[switch],OUTPUT_PORT);
12     Pack(RO[switch],INPUT_PORT);
13   }
14 }
```

Fig. 6. Pseudocode of the path computation phase.

```
1  RegionsComputation(in: Topology T,
2                              RoutingOptions RO[];
3                     out: Regions R[]) {
4    for (switch ∈ T) {
5      output_combinations = GetOutputCombinations(switch);
6      for (input ∈ switch, outputs ∈ output_combinations){
7        destinations = GroupDestinations(input,outputs,
8                                         RO[switch]);
9        if (destinations ≠ ∅)
10         R[switch] = R[switch] ∪ {input,outputs,destinations};
11     }
12   }
13 }
```

Fig. 7. Pseudocode of the region computation phase.

```
1  RegionsMerge(in: Topology T, int max_regions;
2               inout: Regions R[]) {
3    for (switch ∈ T)
4      while (|R[switch]| > max_regions)
5        for (R1,R2 ∈ R[switch])
6          if (CanBeMerged(R1,R2))
7            R[switch] = R[switch] \ {R1} \ {R2} ∪ Merge(R1,R2);
8  }
```

Fig. 8. Pseudocode of the region merge phase.

coming through N port going to destination d can be routed through S output port.

Additionally, routing options may be packed per output port and per input port. First, packing per output port is performed. Two routing options at the same switch are packed if both have the same input port and the same destination ID. For instance, routing options $(N,\ d,\ S)$ and $(N,\ d\ , E)$ can be packed into the routing option $(N, d,\ \{N, E\})$. Notice that by doing this, we are representing adaptive routing options.

Once routing options are packed per output ports, further packing can be done, now depending on the input port. If two routing options at a given switch have the same destination IDs and the same set of output ports, then they can be further packed into one routing option. The input ports of the resulting routing option is the union of all the input ports of the routing options being packed. For instance, in Fig. 6, routing option $(\{N, W, I\}),\ d, \{S, E\})$ has been obtained by packing $(N, d,\ \{S, E\})$, $(W, d\ \{S, E\})$, and $(I, d, \{S, E\})$ routing options.

In the second phase, the algorithm computes the routing regions from the routing options (see Fig. 7). At every switch, the algorithm groups destinations reachable through the same output ports at the switch and coming from the same set of input ports. For instance, Fig. 5 shows some regions computed for the selected switch. In particular, only regions defined for output ports S and E are plotted. The first region $(R1)$ includes switches inside the box defined by switches $x1$ and $x2$. This region is defined for destinations that are reachable only through the S output port and for packets coming through input ports W, N, or I. In the same sense, region $R2$ is defined for packets using output port E and coming through input ports N, W, or I. Finally, region $R3$ is defined for packets coming either from input ports N, W, or I, and can use either output port E or S.

Finally, at the last phase, the algorithm packs all the regions in order to bound the maximum number of allowed regions (see Fig. 8). For this, the algorithm takes a third parameter ($max\_regions$ parameter) that indicates the maximum number of regions allowed at any switch. The algorithm will pack regions by restricting routing. That is, RBR reduces the number of output ports that can be used at a given switch to reach a set of destinations. However, the algorithm guarantees always an output port (i.e., connectivity is ensured).

In order to reduce the number of routing options, the algorithm focuses at switches using a number of regions higher than the maximum allowed ($max\_regions$ parameter). Then, at a given switch, it compares each region with the remaining ones, checking if they can be merged. Two regions can be merged if the combination of both regions defines a box of switches and the output ports of one of the regions is a subset of the output ports of the other region. For instance, from the previous example shown in Fig. 5, the algorithm detects that regions $R2$ and $R3$ can be merged into one region. Effectively, the output port of region $R2(E)$ is a subset of output ports of $R3(\{S, E\})$. However, the resulting region will restrict routing. In order to keep deadlock freedom, the output ports of the resulting region will result from the intersection of the output ports of the two regions being merged. Thus, in the example, the output port of

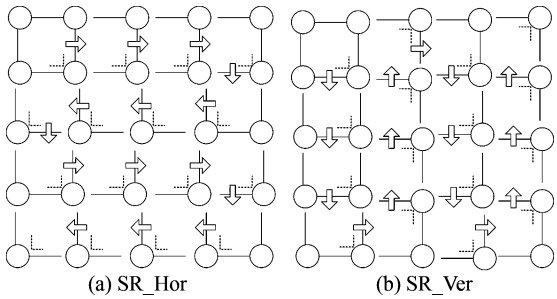


Fig. 9. Different SR segmentation layouts.

the resulting region $R2'$ is E. This way, we prevent the use of the S output port for packets going to $R2'$.

Although routing is restricted, we expect this limitation to have a low impact on final performance. There are two reasons to make this assertion, First, routing is restricted at some particular switches, thus, for a given path, alternative routing options will be eliminated only at some switches. Second, the packing will be performed only at those switches with a high number of regions. As we will see, most of the switches require a very low number of regions, thus not requiring packing.

The complexity of RBR is driven by the processes which compute the routing options and group them to form regions. RBR mechanism uses as an input parameter the topology and the location of the routing restrictions. Thus, RBR complexity does not include the complexity of the routing algorithm. Routing options are computed for every switch and for every combination of input and output ports. For a mesh topology with $N \times N$ nodes the complexity of computing the routing options for a pair of switches is $O(2^N)$.

Once routing options are computed, they are grouped into regions. Regions are obtained by searching routing options for each output port at each switch. Finally, if packing of regions is needed, the algorithm searches for pairs of neighbor regions with shared boundaries. As a summary, the overall cost of RBR is driven by the computation process of regions, thus being $O(N^2 \times d \times 2^N)$, where d is the number of ports per switch. This analysis represents the worst case scenario when the mesh has no routing restrictions. The execution time of the region computation process reduces considerably in the presence of routing restrictions. Heuristic solutions will be necessary to handle large networks.

IV. RBR Applied to a General Purpose Approach

In this section, we provide an analytical evaluation of the number of regions required for different topologies when the application traffic pattern is not known in advance. Thus, any possible pair of end nodes may communicate. We combine RBR with the SR routing algorithm using different segmentation processes. In particular, Fig. 9 shows two different segmentation processes, the first one [see Fig. 9(a)] is referred to as $\mathrm{SR}_{\mathrm{hor}}$ and searches segments from top to bottom and each row in a different direction. In the second one [see Fig. 9(b)] segments are searched from left to right and each column in a different direction. This algorithm is referred to as $\mathrm{SR}_{\mathrm{vert}}$. Fig. 9 shows with arrows the directions taken when performing the segmentation.

For comparison purposes we also show results when combining RBR with the Up*/Down* routing algorithm, either using a breadth-first spanning (BFS) tree and a depth-first spanning (DFS) tree . Also, when applicable, the DOR routing algorithm is evaluated.

TABLE II
Number of Regions Required for Different Regular Topologies and Routing Algorithms

Routing algorithm	min	max
DOR	4	4
SR_{hor}	4	7
SR_{vert}	4	7
BFS	4	6
DFS	4	7

A. Number of Regions Required in a 2-D Mesh

Table II shows the maximum and minimum number of regions required for implementing different routing algorithms in meshes with different size. In the case of minimum number of regions, adaptiveness is minimized. Table II shows that DOR can be implemented with only four regions at maximum on every switch. This is obvious as DOR is deterministic and there are four output ports at each switch. For other routing algorithms, DFS requires six regions, and both SR and UD require seven. However, these routing algorithms are partly adaptive (provide more than one routing option). Indeed, the number of regions for the later routing algorithms can be reduced to only four regions on every switch and independently of network size. Obviously, this reduction may come with an impact on performance. Later, in Section VI, we analyze this. Notice also that the number of regions is constant regardless of network size, as the structure of the network is the same. Therefore, from that point of view the RBR mechanism is scalable.

B. Number of Regions Required in a Faulty Mesh

Different networks have been generated from an 8×8 mesh with up to 10% of different random link failures injected. Such networks help us to analyze the amount of resources (basically the number of regions) required by RBR to guarantee network connectivity. We provide the maximum and minimum (reducing adaptiveness) number of regions required on each topology-routing combination. Results are shown as the percentage of networks that require a particular number of regions.

Fig. 10(a) analyzes all possible combinations of two simultaneous link failures in a 8×8 mesh network. Fig. 10(a) shows the percentage of topologies requiring different number of regions for $\mathrm{SR}_{\mathrm{hor}}$ and $\mathrm{SR}_{\mathrm{vert}}$ with maximum and minimum adaptivity provided. As expected, the minimum and maximum number of regions required increases to 10 and 16, respectively. This is due to the increase in the irregularity of the topology and the need for more regions within the switches. However, 10 regions are enough for supporting 98% of the cases offering full connectivity and minimum adaptivity in $\mathrm{SR}_{\mathrm{hor}}$ and $\mathrm{SR}_{\mathrm{vert}}$. At the same time, 13 regions are required to support maximum adaptivity for the 98% of the cases when using $\mathrm{SR}_{\mathrm{hor}}$ and 96% when using $\mathrm{SR}_{\mathrm{vert}}$.

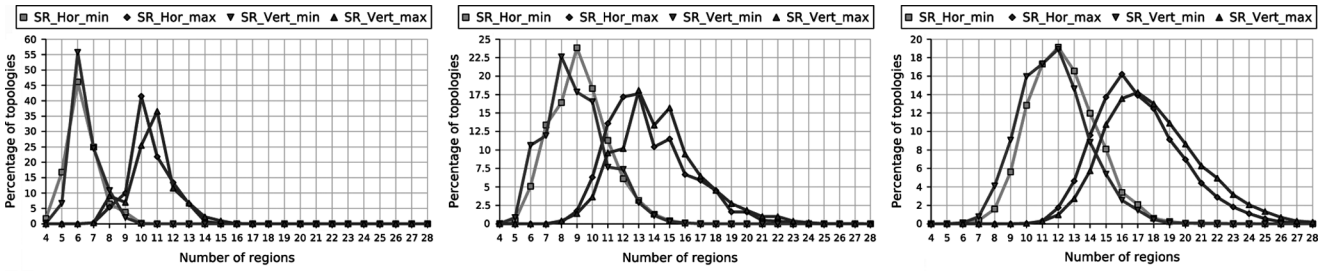


Fig. 10. Percentage of topologies requiring different number of regions with different algorithms: (a) two; (b) four; and (c) ten failures.

We analyzed 12 000 random different combinations of topologies with up to 10 link failures. Selected results for 4 and 10 links failures are shown in Figs. 10(b) and (c), respectively. As it can be noticed, the number of regions required to cope with maximum adaptivity increases with the number of faults. For $\mathrm{SR}_{\mathrm{hor}}$, we need 17 regions when 4 link failures exist and 24 regions when 10 link failures are present. For $\mathrm{SR}_{\mathrm{vert}}$ similar results are obtained ranging from 17 to 28 regions. However, for minimum adaptivity, regions can be packed more efficiently. For $\mathrm{SR}_{\mathrm{hor}}$ regions will vary from 9 (4 links failed) to 16 (10 links failed), whereas $\mathrm{SR}_{\mathrm{vert}}$ ranges from 9 to 17 regions.

To summarize, the number of regions required depends on the regularity of the topology (the number of failures and their position) and the routing algorithm used. At the same time, the number of regions is closely related to the number of routing options. In fact, a way of reducing the number of regions is to limit the adaptiveness of the routing algorithm. There exists a tradeoff between the number of regions and the desired adaptivity represented by the gap between the minimum and maximum number of regions. Such tradeoff is reduced as the irregularity of the topology increases, therefore RBR switches may support a bounded number of faults depending on the number of regions and the desired adaptivity space. As a general conclusion, we can deduce that 16 regions at every switch seems enough to cover all possible cases.

V. Application-Specific Approach

The traditional way to design a routing algorithm leaves out of consideration the characteristics of the communication traffic which will be injected into the network. But now, let us change the perspective by considering the embedded system domain. In this domain the system is usually customized for execution of a particular application (or a limited set of applications). The knowledge of applications which will be mapped on the system has been exploited by several researchers to define new techniques for optimizing several quality indices like performance, power dissipation, and energy consumption [25], [26]. In the context of NoCs, a first attempt to exploit the knowledge of the communication traffic to optimize the routing algorithm has been proposed in [12]. The methodology, named APSRA allows to automatically generate highly adaptive and deadlock-free routing algorithms tailored for a specific application and network topology.

In this section, we briefly describe the APSRA design methodology, and then evaluate the number of regions required when implemented with RBR.

A. APSRA Design Methodology

After the task mapping phase of the NoC design flow, we have a complete knowledge about the pairs of cores which communicate and other pairs which never communicate. This information is captured by means of a *communication task graph* (CG). The CG is a direct graph where each vertex represents a computational module in the application (i.e., a task) and each directed arc between two tasks characterizes either data or control dependencies. The communication graph represents the first input of the APSRA methodology. The second input of the methodology is the *topology graph* (TG). The TG is a direct graph where each vertex represents a network node (either a core or a switch), and each directed arc between two nodes represents an unidirectional channel.

The methodology starts assuming a fully adaptive routing function. Based on it, the channel dependency graph (CDG) [27] is built. In the second step, the *Application Specific Channel Dependency Graph* (ASCDG) is extracted from CDG. The ASCDG is a sub-graph of CDG in which the sub-set of the direct dependencies are generated by the routing paths connecting only the communicating network nodes.

If the ASCDG is not acyclic, in the third step, all the cycles are iteratively broken. To break a cycle, at least one of the dependencies which forms the cycle has to be removed. This step consists of two phases. In the first phase, the dependency to be removed is selected. In [12], an heuristic was presented to select the dependency in such a way that the impact on adaptivity is minimized. In the second phase, the dependency is removed simply by restricting the routing function.

Finally, in the fourth step, the routing tables are populated by using the information provided by the CG and the routing function.

RBR can be generalized to support communication information as presented in [8]. By using the information of nodes that never communicate the compression level is generally increased since the process of region packing can be limited only to the subset of destinations associated to a particular node.

For example, Fig. 11(a) shows the computed regions (as discussed in Section III-C) for the switch plotted in a black circle before packing the regions. By using the information stored in the communication graph, it is possible to compute only the

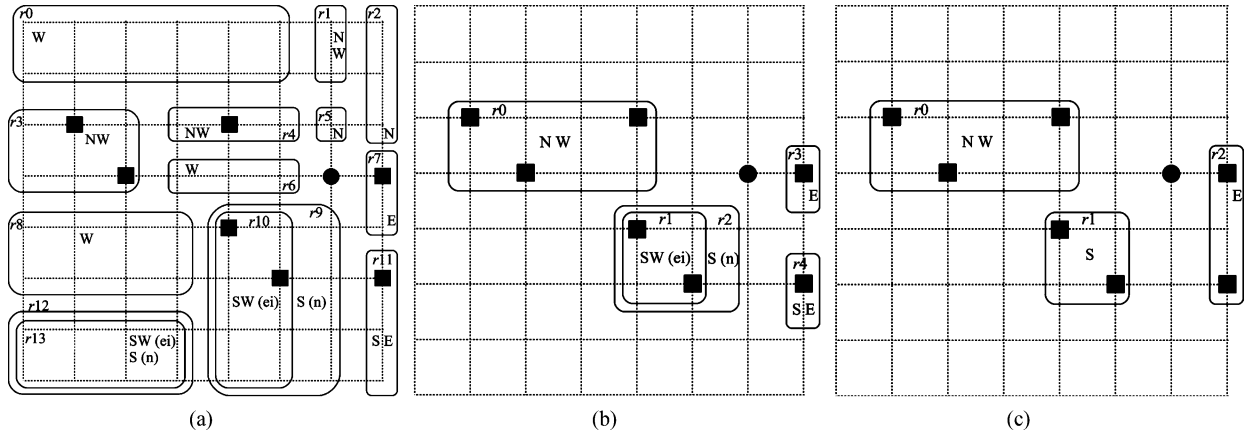


Fig. 11. RBR generalized to take advantage of communication information. Regions for switch (3,6). (a) When all the network nodes are considered as possible destinations. (b) When only actual destinations are considered. (c) After maximum packing.

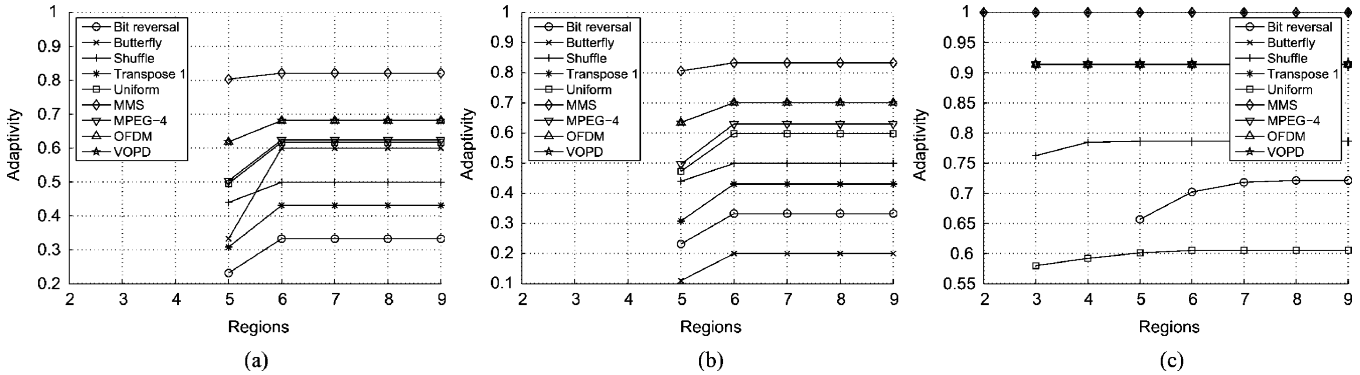


Fig. 12. Variation of adaptivity for different number of regions under different traffic scenarios. (a) $\mathrm{SR}_{\mathrm{hor}}$; (b) $\mathrm{SR}_{\mathrm{vert}}$; and (c) APSRA. We use the definition of adaptivity (sometimes also referred as degree of adaptiveness) given by Glass and Ni in [13].

destination nodes which will be reached through such switch. Let us suppose that such destinations are those plotted in the figure with a black square. The packetization process can thus be performed on just the regions containing the actual destination nodes [see Fig. 11(b)]. The number of required regions is thus reduced if communication information is added to RBR. As it can be observed from Fig. 11(c), the number of required regions after packetization reduces to three regions as compared to five regions when communication information is ignored.

B. Traffic Scenarios

For evaluation purposes we use two sets of traffic scenarios. In the first set, synthetic traffic scenarios (bit reversal, butterfly, shuffle, transpose and uniform patters) are considered. The traffic scenarios belonging to the second set have been captured from the following real applications.

- *MMS*: A generic MultiMedia System which includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder, and an mp3 audio decoder [28]. The application is partitioned into 40 distinct tasks and then these tasks were assigned and scheduled onto 25 selected IPs.
- *OFDM*: A MIMO-OFDM receiver which is mapped on 16 IPs as described in [29]. To support the maximum data rate of world-wide spectrum efficiency (WWiSE) proposal for the next-generation wireless LAN systems, some of IPs have been parallelized to multiple IPs.
- *MPEG-4*: A MPEG-4 decoder mapped on 12 IPs as described in [30]. In this application, many of the IPs communicate with each other through a shared SDRAM. This makes this traffic similar to the hot-spot traffic.
- *VOPD*: A Video Object Plane decoder mapped on 12 IPs as described in [30]. Here, half or the IPs communicate to more than a single IP.

All the previously described applications have been mapped on a 8×8 mesh-based NoC architecture by using the topological mapping algorithm presented in [31]. As the number of IPs is lower than the number of tiles, a set of tasks generating uniform communication traffic patterns have been randomly mapped in the remaining unallocated tiles of the mesh.

C. Number of Regions Required for Varying Traffic Classes

Under the hypothesis of a regular design in which all the switches of the NoC are the same (i.e., they are multiple instances of a reference switch design which implements the same number of regions), Fig. 12 shows the number of regions each switch must implement to cover the source-destination paths defined by the original routing table and the communication graph. For each routing algorithm and for each traffic scenario, the graphs show the variation of adaptivity for different region budgets. As can be observed, for all the traffic scenarios considered, a switch implementing six regions is able to manage both SR and APSRA routing algorithms without any appreciable loss in

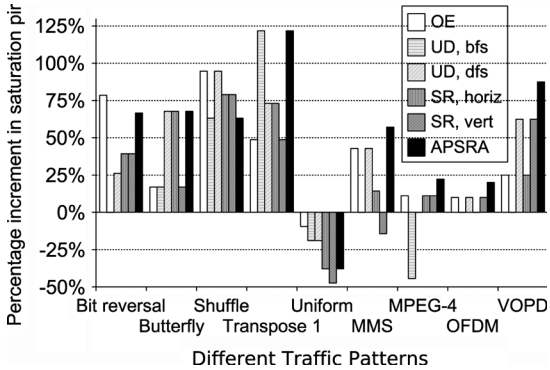


Fig. 13. Percentage increment in saturation *pir* using DOR as baseline.

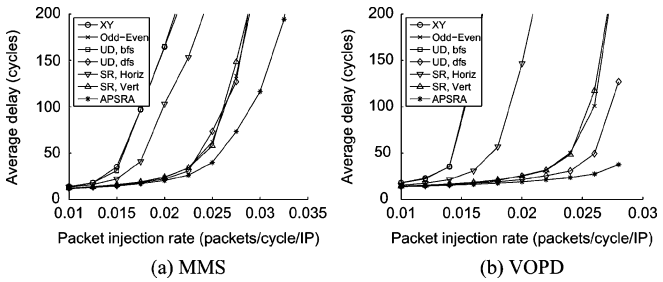


Fig. 14. Delay variation under (a) *MMS* traffic and (b) *VOPD* traffic.

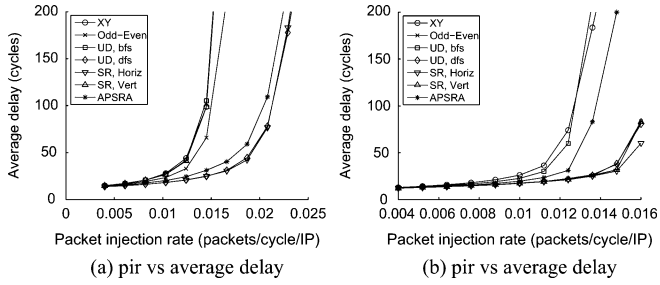


Fig. 15. Delay variation under (a) *Butterfly* and (b) *Shuffle* traffic.

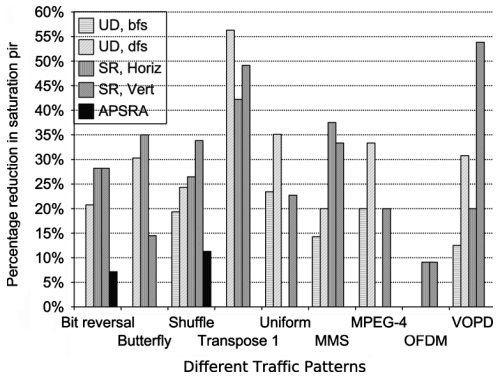


Fig. 16. Percentage reduction in saturation *pir* when maximum compressed routing tables are used.

adaptivity. For APSRA, in particular, four regions are enough with exception of *bit-reversal* and *uniform*. In fact, the high degree of adaptiveness which characterizes APSRA translates in more chances of using the same region to incorporate more destinations. It is possible, however, to reduce the cost of the switch by lowering the number of regions down to two. In this case, some routing restrictions have to be applied with a consequent negative impact on adaptivity. The decrease in adaptivity ranges from 2% to 5% for APSRA and from 3% to 27% for SR (both $\mathrm{SR}_{\mathrm{hor}}$ and $\mathrm{SR}_{\mathrm{vert}}$).

VI. Performance Evaluation of APSRA and SR When Using RBR

In this Section, we evaluate performance of different routing algorithms using *Noxim* [32], a flit-accurate NoC simulator developed in SystemC. As performance metrics, we use *throughput* (TP) and *average delay* (D).

We performed the experiments on 8×8 networks using wormhole switching with a constant packet size of 8 flits. The maximum bandwidth at each link was set to 1 flit per cycle. Switches have an input-buffer size of 4 flits and random selection policy at the arbiter for adaptive routing algorithms. For synthetic traffic, we use Poisson packet injection distribution while for the real traffic scenarios, we use self-similar packet injection distribution as it has been observed in the bursty traffic between on-chip modules in typical multimedia and networking applications [33]. We evaluated the full range of traffic density, from low load to saturation. For each load value, latency values were averaged over 60 000 packet arrivals after a warm-up session of 30 000 arrived packets. The 95% confidence intervals are mostly within 2% of the means.

We conducted the evaluation in two separate scenarios. First, we compared different routing algorithms (odd-even OE, BFS, DFS, $\mathrm{SR}_{\mathrm{hor}}$, $\mathrm{SR}_{\mathrm{vert}}$, APSRA) with an unbounded number of regions (up to 8 regions are enough for the 8×8 network used) and without compression (packing) of the routing options. Here, we obtain the performance achieved by the algorithms when they offer their maximum possible adaptivity. Then, in a second scenario, we bound the number of regions down to four, thus, decreasing the adaptivity provided by the algorithms to their minimum in order to analyze how much performance is lost.

A. Unbounded Comparison

The described routing algorithms are analyzed under different traffic scenarios in the 8×8 mesh network with no link failures. In this scenario, our switch model uses an unbounded number of regions (eight are enough) in order to achieve the maximum network adaptivity offered by the routing algorithms.

Fig. 13 presents the percentage of increase in *saturation packet injection rate* (*pir*) achieved by the different routing algorithms with respect to the results obtained by the DOR algorithm under different traffic patterns. The saturation *pir* is the value of the packet injection rate at which an increase in the applied load does not result in linear increase in throughput [34]. We calculate the saturation *pir* as the packet injection rate at which throughput stops increasing linearly with *pir* and its slope drops more than 5% from the previous averaged slope.

As we can observe in Fig. 13 APSRA behaves very well in real traffic applications (MMS, MPEG-4, OFDM, VOPD) achieving for VOPD traffic up to 25% of increase in saturation *pir*. Note that APSRA is the only routing algorithm designed

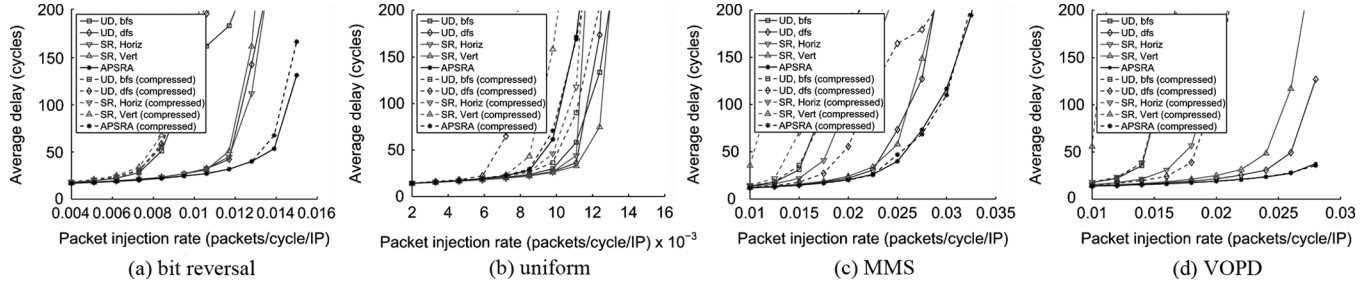


Fig. 17. Delay variation for uncompressed (solid line) and maximum compressed (dashed line) routing tables under (a) *Bit reversal* traffic, (b) *Uniform* traffic, (c) *MMS* traffic, and (d) *VOPD* traffic.

from the communication graph. Fig. 14 shows the delay characteristics for the real traffic scenarios considered. In fact, APSRA obtains minimum average delay in all cases with VOPD traffic being the one that benefits the most. Another observation from Fig. 13 is that DOR achieves higher throughput than the rest of routing algorithms for the uniform traffic pattern. This result is well known as DOR behaves even better than fully adaptive routing algorithms in regular meshes. However, for other traffic patterns DOR obtains significantly lower network throughput. It can be noticed that DFS, $\mathrm{SR_{hor}}$, $\mathrm{SR_{vert}}$, and OE benefit from their adaptiveness for bit-reversal, shuffle, and transpose patterns (also, $\mathrm{SR_{hor}}$ and DFS achieve higher network throughput for the butterfly pattern).

Fig. 15 shows the delay and throughput variation for different values of packet injection rate under *butterfly* and *shuffle* traffic. As it can be observed, SR and UD perform better than APSRA in terms of average delay and throughput. This is an interesting result. In fact, APSRA offers full adaptivity for this traffic patterns (degree of adaptiveness equal to one). However, from the average delay point of view APSRA does not perform so well. This can be seen as an informal proof that high adaptivity does not always mean better performance. Based on this observation, we predict that RBR will not suffer significant performance degradation with the loss in adaptivity due to the reduction in the number of regions. In the next section, we test this hypothesis.

B. Bounded Comparison

In Section IV, we quantified the minimum number of regions required to provide deadlock freedom and full connectivity for different routing algorithms. As stated, four regions are enough to guarantee deadlock freedom and full connectivity in regular meshes. In this section, we bound the number of regions within a switch to four in order to analyze how the reduction translates into performance penalties.

Fig. 16 presents the percentage of the reduction in *saturation pir* for all traffic scenarios when maximum compressed routing tables are used. It is interesting to note that there is not appreciable performance degradation for APSRA in almost all the traffic patterns (synthetic and real applications). Indeed, APSRA has been designed to achieve maximum optimization of the routing options. It requires few regions and performs very well in comparison to the other routing algorithms. UD routing does not present significant drop in performance (less than 30%) due to the poor performance presented in the previous unbounded scenario. As it is well known,

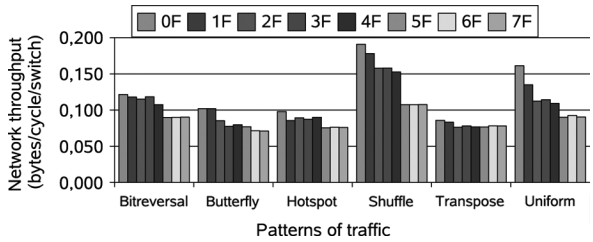


Fig. 18. Network throughput. 8×8 mesh with an increasing number of failures. SR with eight regions.

UD tends to saturate very early as it concentrates the traffic around the root node. Both SR and DFS suffer a significant drop in network throughput for *uniform*, *bit-reversal*, *shuffle*, and *transpose* traffic patterns. This is due to the reduction in adaptivity. We need to bear in mind that this is the worst case scenario where the algorithms acquire a totally deterministic behavior.

This fact is confirmed in Fig. 17 where we can appreciate the variation of the average delay for the most representative synthetic (*bit-reversal* and *uniform*) and real (MMS and VOPD) traffic scenarios. We can observe here how DFS and both cases of SR loose up to 40% of its throughput. To summarize, when using minimum number of regions performance of general purpose routing algorithms is considerably reduced due to reduction in routing options, but APSRA maintains its performance intact due to the nature of its design.

Let us now analyze the degradation in performance as we increase the number of failures in the network. The goal of this final analysis is to determine the number of regions required to achieve acceptable levels of performance and fault tolerance. In Section IV, we also explored the number of regions required to tolerate up to ten link failures while providing some adaptivity. As we stated early, with 16 regions we are able to tolerate up to 7 link failures offering minimum adaptivity and full connectivity for the 99% of cases using $\mathrm{SR_{hor}}$ and $\mathrm{SR_{vert}}$.

Fig. 18 shows the throughput degradation with an increasing number of link failures for different traffic patterns [throughput achieved in a mesh network with no link failures (0F) is included]. Our switch is provided with just eight regions, this case is within the 40% of cases that achieve minimum adaptivity with eight regions. As the number of failures increases, the number of routing options eliminated to keep the number of regions constant increases, thus, performance decreases due to the irregularity of the network and the decrease of adaptiveness. As we

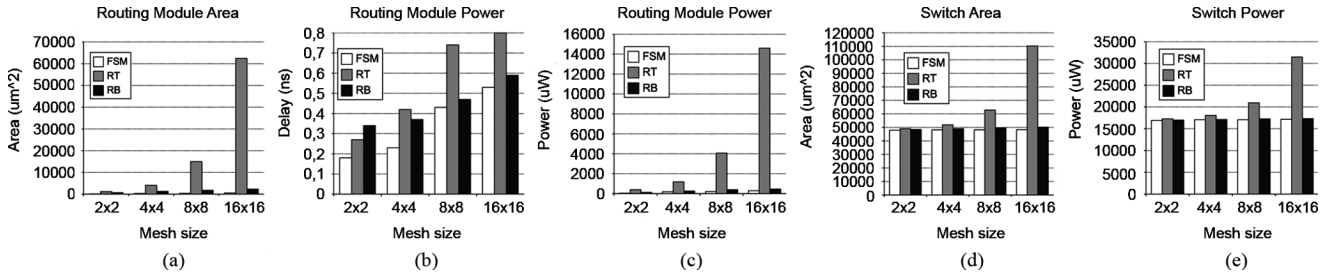


Fig. 19. (a) Area, (b) delay, and (c) power of the block implementing the routing table (TB) and the region-based mechanism (RB). (d) Area and (e) power of the entire switch.

can see, the performance degradation for hot-spot and transpose traffic is less than 25%. For the rest of patterns, throughput decreases 30% for bit-reversal and uniform, 25% for butterfly, and 40% for shuffle. Therefore, as it was expected, the performance of the network decreases as the number of failures increase up to the point that network throughput becomes constant.

It is worth to note some unexpected performance variations as the number of failures increase. For example, in hotspot traffic, configurations with 2 and 4 failures achieve slightly better results than configurations with 1 and 3 failures. These results are due to the way the routing algorithm is reconfigured, some routing restrictions are changed and also different adaptive routing options are considered. In these particular cases, the presence of these new failures help to balance better the traffic across the network. The overall throughput degradation, however, is consistent.

C. Area and Energy Cost Evaluation

Routing with forwarding tables requires a table with as many entries as the number of nodes and input ports, even more, every entry needs to store the different number of ports returned by the routing function. Hence, the cost of this alternative is $N \times d \times d$, where N is the number of nodes and d is the number of ports. However, if we consider mesh topologies and minimal routing, the set of admissible output ports returned by the routing function has cardinality of two at maximum. Therefore, in this case, the routing table has a cost of $N \times 2 \times d$ bits. As it can be seen, for memory-based solutions the overall requirement of memory grows linearly with the network size. Contrary to this, the region-based routing requires per region no more than 30 gates, four registers of size $\lceil \log_2(N)/2 \rceil$ (ROW_1, COL_1, ROW_2, and COL_2 registers), one register with $d+1$ bits (IP register) and a register with d bits (OP register).

The modules implementing the routing function in an FSM-based switch (FSM), in a table-based switch (TB), and in a region-based switch (RB) have been designed and synthesized using Synopsys Design Compiler and mapped on a 90-nm technology library from TSMC. The FSM-based switch implements the DOR routing algorithm. 16 regions were used for the region-based switch and a table of $N \times 2 \times d$ bits for the table-based implementation. In both scenarios, we considered 64-bit, 4-flits FIFO input buffers. The area, delay, and power figures of the three switches for different mesh sizes are shown in Fig. 19(a)–(c). As expected, TB is much more expensive than FSM and RB both in terms of area and power dissipation. For instance, for an 8×8 mesh, RB requires $8\times$ less area and $10\times$ less power than TB. However, in terms of delay, the advantages

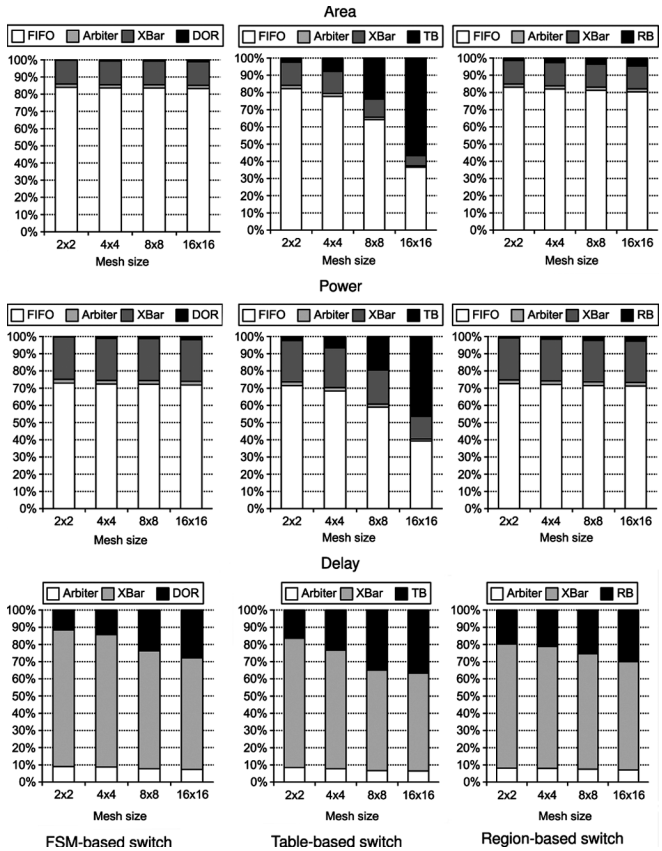


Fig. 20. Breakdown of area, power, and delay for different mesh sizes.

of RB are evident for mesh sizes greater than 4×4. Comparing FSM with RB, with the exception made for the area which increases by a factor 3, RB does not exhibit important overhead. Delay and power increase less than 1% and 8%, respectively, which can be considered low enough if compared with the overall performance and flexibility improvement provided by RB. However, it should be pointed that such overheads refer to the module implementing the routing function only and not to the entire switch. Fig. 19(d) and (e) show area and power dissipation of the entire switch. As it can be observed, both the area and power weight of a region-based switch are close to that of an FSM-based switch. For instance, for a 16×16 mesh, in terms of silicon area, a region-based switch is less than 4% more expensive than an FSM-based switch and more than 55% cheaper than a table-based switch. In terms of power consumption, a region-based switch is less than 1% more expensive than an FSM-based switch and more than 45% cheaper than a table-based switch.

Fig. 20 shows the breakdown of area, power, and delay in the switch. As it can be observed, the routing module affects the overall area and power budget by only a few percent. FIFO buffers and crossbar dominate the design both in terms of area and power consumption when routing table implementation is used in small meshes, but while increasing the size of the network, routing tables start to represent a major contribution. For example, for a 16×16 mesh, routing tables contribute almost 60% of the switch silicon area, and 50% of the switch power. Although the critical path is dominated by the latency of the crossbar, the routing function computation (performed by means of either routing table or using the region concept) starts also to be significant with the increase of the mesh size. For a 16×16 mesh such contribution reaches 40% for the table-based switch and only 30% and 28% for the region-based implementation and FSM implementation, respectively. Finally, the contribution in both silicon area and power dissipation of the block implementing the region-based routing is negligible as compared with the FIFO buffers and crossbar. In addition, they are quite insensitive to the network size. All of this clearly shows the benefits in latency, power optimization and area that the region-based switch brings for large networks when compared to direct table-based solutions.

VII. Conclusion

In this paper, we have proposed an efficient scheme to reduce the memory requirements in table-based switches for NoCs. RBR reduces area requirements and makes the option of table-based switch design practical. RBR can be applied to any topology and any routing algorithm. Indeed, we have applied RBR to two distinct routing algorithms, namely SR and APSRA, in regular mesh networks and irregular topologies (derived from mesh networks).

Evaluation results show that RBR keeps logic requirements almost independent on network size. The number of regions required by RBR depends on the regularity of the topology, the number of routing options (degree of adaptivity) and the routing algorithm used. With only four regions several routing algorithms for a 2-D mesh can be applied with no performance degradation. By using up to 16 regions in a 8×8 mesh we are able to tolerate up to 7 link failures offering for 99% of all the cases. Moreover, latency and throughput performance is not affected when APSRA is combined with RBR. Finally, for large networks, RBR achieves the same performance ass table-based solutions with much less area requirements and power consumption levels.

References

[1] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc. Conf. Des., Autom. Test Eur. (DATE)*, NY, 2000, pp. 250–256.

[2] S. Kumar, A. Jantsch, M. Millberg, J. Oberg, J.-P. Soininen, M. Forsell, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proc. ISVLSI*, 2002, p. 0117.

[3] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Proc. 20th Ann. Int. Conf. Supercomput. (ICS)*, NY, 2006, pp. 187–198.

[4] E. Bolotin, A. Morgenshtein, I. Cidon, R. Ginosar, and A. Kolodny, "Automatic hardware-efficient soc integration by qos network on chip," in *Proc Electron, Circuits Syst. (ICECS)*, 2004, pp. 479–482 [Online]. Available: citeseer.ist.psu.edu/dally01route.html

[5] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Qnoc: Qos architecture and design process for network on chip," *J. Syst. Arch.*, vol. 50, no. 2–3, pp. 105–128, 2004.

[6] T. Bengtsson, S. Kumar, R. Ubar, and A. Jutman, "Off-line testing of crosstalk induced glitch faults in noc interconnects," in *Proc. 24th Norchip Conf.* , Linkoping, Sweden, 2006, pp. 221–225.

[7] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Mateo, CA: Morgan Kaufman, 2004.

[8] M. Palesi, S. Kumar, and R. Holsmark, "A method for router table compression for application specific routing in mesh topology NoC architectures," in *Proc. SAMOS*, 2006, pp. 373–384.

[9] J. Flich, A. Mejia, P. Lopez, and J. Duato, "Region-based routing. An efficient routing mechanism to tackle unreliable hardware in network on chips," presented at the 1st Int. Symp. Netw.-on-Chips, Princeton, NJ, May 2007.

[10] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Routing table minimization for irregular mesh NoCs," in *Proc. Conf. Des., Autom. Test Eur. (DATE)*, NY, 2007, pp. 942–947.

[11] A. Mejia, J. Flich, J. Duato, S. Reinemo, and T. Skeie, "Segment-based routing: An efficient fault-tolerant routing algorithm for meshes and tori," presented at the 20th Int. Parallel Distrib. Process. Symp. (IPDPS), Rhodos, Grece, Apr. 2006.

[12] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A methodology for design of application specific deadlock-free routing algorithms for NoC systems," in *Proc. 4th Int. Conf. Hardw./Softw. Codes. Syst. Synth. (CODES+ISSS)*, NY, 2006, pp. 142–147.

[13] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *J. ACM*, vol. 41, no. 5, pp. 874–902, 1994.

[14] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 729–738, Jul. 2000.

[15] T. Skeie, O. Lysne, and I. Theiss, "Layered shortest path (lash) routing in irregular system area networks," in *Proc. Commun. Arch. Clusters*, 2002, pp. 162–169.

[16] J. Flich, P. Lopez, J. Sancho, A. Robles, and J. Duato, "Improving infiniband routing through multiple virtual networks," in *Proc. 4th Int. Symp. High Perform. Comput. (ISHPC)*, London, U.K., 2002, pp. 49–63.

[17] O. Lysne and T. Skeie, "Load balancing of irregular system area networks through multiple roots," in *Proc. Int. Conf. Commun. Comput.*, Jun. 2001, pp. 165–171.

[18] T. Skeie, O. Lysne, J. Flich, P. Lopez, A. Robles, and J. Duato, "Lashtor: A generic transition-oriented routing algorithm," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2004, pp. 595–604.

[19] M. Koibuchi, A. Jouraku, K. Watanabe, and H. Amano, "Descending layers routing: A deadlock-free deterministic routing using virtual channels in system area networks with irregular topologies," in *Proc. Int. Conf. Parallel Process. (ICPP)*, Oct. 2003, pp. 527–536.

[20] L. Cherkasova, V. Kotov, and T. Rokicki, "Designing fibre channel fabrics," in *Proc. Int. Conf. Comput. Des. (ICCD)*, Washington, DC, 1995, p. 346.

[21] J. C. Sancho, A. Robles, and J. Duato, "New methodology to compute deadlock-free routing tables for irregular networks," in *Proc. Workshop Commun., Arch., Appl. Netw.-Based Parallel Comput. (CANPC)*, Jan. 2000, pp. 45–60.

[22] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, and T. L. Rodeheffer, "Autonet: A high-speed, self-configuring local area network using point-to-point links," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 8, pp. 1318–1335, Oct. 1991.

[23] J. Sancho, A. Robles, and J. Duato, "A flexible routing scheme for networks of workstations," in *Proc. 3rd Int. Symp. High Perform. Comput. (ISHPC)*, London, U.K., 2000, pp. 260–267.

[24] M. Koibuchi, A. Funahashi, A. Jouraku, and H. Amano, "L-turn routing: An adaptive routing in irregular networks," in *Proc. Int. Conf. Parallel Process. (ICPP)*, Washington, DC, 2001, pp. 383–392.

[25] G. Ascia, V. Catania, and M. Palesi, "A multi-objective genetic approach for system-level exploration in parameterized systems-on-a-chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 4, pp. 635–645, Apr. 2005.

[26] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Architectures and synthesis algorithms for power-efficient bus interfaces," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 9, pp. 969–980, Sep. 2000.

[27] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 547–553, May 1987.

[28] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, Apr. 2005.
[29] S.-R. Yoon, J. Lee, and S.-C. Park, "Case study: NoC based next-generation wlan receiver design in transaction level," in *Proc. Int. Conf. Adv. Commun. Technol.*, Feb. 2006, vol. 2, pp. 1125–1128.
[30] K. Srinivasan and K. S. Chatha, "A technique for low energy mapping and routing in network-on-chip architectures," in *Proc. Int. Symp. Low Power Electron. Des.*, San Diego, CA, 2005, pp. 387–392.
[31] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," in *Proc. 2nd IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codes. Syst. Synth.*, Stockholm, Sweden, Sep. 2004, pp. 182–187.
[32] Sourceforge.net, "Noxim: Network-on-chip simulator," 2008. [Online]. Available: http://noxim.sourceforge.net
[33] G. Varatkar and R. Marculescu, "Traffic analysis for on-chip networks design of multimedia applications," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2002, pp. 510–517.
[34] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.
[35] T. T. Ye, L. Benini, and G. D. Micheli, "Packetization and routing analysis of on-chip multiprocessor networks," *J. Syst. Arch.*, vol. 50, no. 2–3, pp. 81–104, 2004.
[36] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Neighbors-on-path: A new selection strategy for on-chip networks," in *Proc. IEEE/ACM/IFIP Workshop Embed. Syst. Real Time Multimedia*, Seoul, Korea, 2006, pp. 79–84.
[37] A. S. Vaidya, A. Sivasubramaniam, and C. R. Das, "Lapses: A recipe for high performance adaptive router design," in *Proc. HPCA*, 1999, p. 236.

Andres Mejia received the M.S. degree in electronic engineering from the Pontificia Bolivariana University Colombia, in 2000, and is pursuing the Ph.D. degree in the computer engineering from Universidad Politécnica de Valencia, Valencia, Spain.

He is a Research Assistant with the Department of Computer Engineering, Universidad Politécnica de Valencia. He has worked with different companies including Nortel Networks, U.K. and Sun Microsystems. His research interests include high-speed interconnects, on-chip networks, multiprocessor architectures, cluster architectures, and routing algorithms.

Mr. Mejia is a student member of the IEEE Computer Society.

Maurizio Palesi received the Laurea degree and the Ph.D. degree in computer engineering from University di Catania, Catania, Italy, in 1999 and 2003, respectively.

Since December 2003, he has held a research contract as an Assistant Professor with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Facolt di Ingegneria, University di Catania. His research interests focus on platform-based system design, design space exploration, low-power techniques for embedded systems, and network-on-chip architectures.

José Flich received the M.S. and Ph.D. degrees in computer science from the Technical University of Valencia (Universidad Politécnica de Valencia), Valencia, Spain, in 1994 and 2001, respectively.

He joined the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia, in 1998, where he is currently an Associate Professor of computer architecture and technology. He has served as Program Committee member in different conferences, including ICPP, IPDPS, HiPC, CAC, ICPADS, and ISCC. He is currently cochair of CAC and INA-OCMC workshops and vice-chair (high-performance networks track) of EuroPar conference. His research interests are related to high performance interconnection networks for multiprocessor systems, cluster of workstations, and networks-on-chip.

Shashi Kumar (SM'08) received the B.Tech., M.Tech., and Ph.D. degrees from the Indian Institute of Technology Delhi, Delhi, India, in 1974, 1976, and 1985, respectively.

He is a Professor with the Department of Embedded Systems, School of Engineering, Jönköping University, Jönköping, Spain. His research interests include system-level modeling and synthesis, parallel architectures and algorithms, reconfigurable computing, and heuristic search algorithms. He was a member of the team which was the first to propose the idea of packet switched communication for on-chip communication and coined the term network-on-chip (NoC) in 2000. His research interests include various aspects of NoC design including NoC topologies, QoS issues in NoC communication, NoC architectural modeling and evaluation, application specific NoC architecture design, mapping applications to NoC platforms, and testing of NoC.

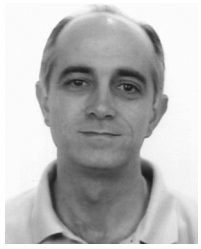
Pedro López received the B.Eng. degree in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Universidad Politécnica de Valencia, Valencia, Spain, in 1984, 1990, and 1995, respectively.

He is a full Professor in computer architecture and technology with the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia. He has taught several courses on computer organization and architecture. His research interests include high performance interconnection networks for multiprocessor systems and cluster of workstations. He has published more than 100 refereed conference and journal papers.

Dr. López is a member of the editorial board of Parallel Computing journal. He is a member of the IEEE Computer Society.

Rickard Holsmark received the Bachelor of Science degree in electronics, with specialization in microcontroller systems and the Master of Science degree in electronics, with specialization in embedded systems from Jönköping University, Jönköping, Sweden, in 2001 and 2003, respectively, where he is currently pursuing the Ph.D. degree in electronic system design.

His research interests focus on specialized architectures and routing algorithms for networks-on-chip. His other areas of interest include embedded systems in general, system level design and processor architectures.



José Duato (M'95) received the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic University of Valencia, Valencia, Spain, in 1981 and 1985, respectively.

He was an Adjunct Professor with the Department of Computer and Information Science, The Ohio State University, Columbus. He is currently a Professor with the Department of Computer Engineering (DISCA), Polytechnic University of Valencia. His research interests include interconnection networks and multiprocessor architectures. He has published more than 380 refereed papers. He proposed a powerful theory of deadlock-free adaptive routing for wormhole networks. Versions of this theory have been used in the design of the routing algorithms for the MIT Reliable Router, the Cray T3E supercomputer, the internal router of the Alpha 21364 microprocessor, and the IBM BlueGene/L supercomputer. He is the first author of *Interconnection Networks: An Engineering Approach* (Morgan Kaufmann, 2002).

Dr. Duato was a member of the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, and *IEEE Computer Architecture Letters*. He was a general cochair of ICPP'01, the program committee chair of HPCA, and a program cochair of ICPP He was also a cochair, a member of the steering committee, the vice-chair, or a member of the program committee for more than 55 conferences, including the most prestigious conferences in his area: HPCA, ISCA, IPPS/SPDP, IPDPS, ICPP, ICDCS, EuroPar, and HiPC.